WHAT IS CLAIMED IS:

1. A method of debugging code that executes in a multithreaded processor having a plurality of microengines comprises:

inserting a segment of executable code into an unused section of a target microengine's microstore in response to a first context swap of one of a plurality of hardware-supported execution threads of a program executing in the target microengine;

executing the segment of executable code; and

resuming execution of the program in response to a second context swap.

2. The method of claim 1 wherein inserting comprises:

saving program counters associated with the plurality of hardware-supported execution threads;

modifying the target microengine's program counters to jump to a start of the segment of executable code; and

appending the saved program counters to an end of the segment of executable code.

3. The method of claim 1 wherein inserting further comprises receiving a user request to execute the segment of executable code.

4. The method of claim 1 wherein executing further comprises examining states of the execution threads at the second context swap.

5. The method of claim 1 wherein resuming further comprises removing the segment of executable code from the microstore.

6. The method of claim 1 wherein resuming comprises restoring the programs counters of the plurality of hardware-supported execution threads that have not executed.

7. The method of claim 1 wherein the segment of executable code resides in a library executable code segments residing in the processor.

8. A method of debugging software that executes in a multithreaded processor having a plurality of microengines comprises:

pausing program execution in a plurality of threads of execution within a target microengine;

inserting a segment of executable code into an unused section of the target microengine's microstore;

executing the segment of executable code in the target microengine; and

resuming program execution in the target microengine.

9. The method of claim 8 wherein pausing is in response to a user command to pause.

10. The instruction of claim 8 wherein the user command to pause further comprises selecting the target microengine from one of the plurality of microengines.

11. The method of claim 10 wherein the user command to pause further comprises selecting the segment of executable code.

12. The method of claim 8 wherein pausing further comprises determining when one of the plurality of threads of execution context swaps.

13. The method of claim 8 wherein inserting further comprises:

modifying a program counter of the paused program to point to the segment of executable code; and

modifying a program counter at an end of the segment of executable code to point to the paused program.

14. The method of claim 8 wherein the segment of executable code causes the target microengine to write to specific registers.

15. The method of claim 14 wherein the specific registers are examined by a user during execution of the segment of executable code.

16. A processor that can execute multiple contexts and that comprises:

a register stack;

a program counter for each executing context;

an arithmetic logic unit coupled to the register stack and a program control store that stores a breakpoint command that causes the processor to:

pause program execution in a context in the processor;

insert a segment of executable code into an used section of a microstore associated with the context;

execute the segment of executable code; and

resume program execution.

17. The processor of claim 16 wherein program execution is paused by disabling a processor enable bit.

18. The processor of claim 16 wherein the segment is inserted in response to a user request received through a remote user interface connected to the processor.

19. The processor of claim 16 wherein an end of the segment pints to a program counter of the program.

20. The processor of claim 16 wherein the segment examines states of execution of the contexts.

21. The processor of claim 16 wherein program execution is resumed by enabling a processor enable bit.

22. A computer program product, disposed on a computer readable medium, the product including instructions for causing a multithreaded processor having a plurality of microengines to:

pause program execution in a plurality of threads of execution within a target microengine;

insert a segment of executable code into an unused section of the target microengine's microstore;

execute the segment of executable code in the target microengine; and

resume program execution in the target microengine.